ABSTRACT

This paper describes the methodology used to compute life-cycle costs of military computer systems as a function of three competing Military Computer Family Architecture candidates (the IBM S/370, DEC PDP-II, and Interdata 8/32), and it presents separate results of applying this methodology to two different models (called Top-down and Bottom-up) of computer resource requirements in the military. The architecture comparisons are made by projecting, computing and combining estimates of hardware and software costs with architecture-dependent factors to obtain life-cycle (development plus maintenance) costs and cost ratios. The results indicate that the three architectures have roughly comparable life cycle costs for the two models. However, neglecting the uncertainties of the input data and assumptions, the bottom-up results indicate that in most circumstances the DEC PDP-II is superior to both the IBM S/370 and Interdata 8/32 architectures. The top-down model results, on the other hand, indicate that the S/370 is superior for high (greater than one) software-to-hardware cost ratios, the Interdata 8/32 is slightly better for low (less than one-fourth) ratios, and the PDP-II is best in between.

INTRODUCTION

Background

The motivation for this work was to help the Army and Navy determine the most cost effective of three candidate computer architectures, with the intent that this architecture would form the basis of a software-compatible Military Computer Family (MCF) over at least the next decade. The three candidate architectures were the IBM S/370, the Digital Equipment Corporation PDP-II, and the Interdata 8/32. It was felt that inclusion of cost considerations would be a vital element in the acceptance of the Selection Committee's recommendations by DoD higher management.

Overview

The life-cycle cost evaluations described are based on (1) a methodology for combining the results of architectural attribute evaluations (i.e., processor and storage efficiency, support-software availability) with computer resource requirements (i.e., bits of storage, processor speed, program sizes) to compute dollar costs of these resources as a function of architecture, and (2) models defining these computer resource requirements considered to be representative of military tactical and strategic systems in the next 10 to 15 years.

Architecture Attributes

The CFA Selection Committee generated the architectural attribute evaluation results that are the basic inputs to the cost computations. The first of these are the data derived from test program experiments. These data indicate the relative efficiencies of the architectures in utilizing storage and processor hardware resources. The "S" measure is a count of the number of storage bytes required to contain programs, given an architecture. Differences in S can be directly related to differences in the amount of
storage and therefore cost requirements. The "M" measure is a count of the number of bytes transferred between the CPU and main memory (including cache) during execution of the test programs, and the "R" measure is a count of the number of bytes transferred among the registers of the CPU. M and R are clearly indicators of the hardware bandwidth requirements of an architecture to do a job. Everything else being equal, memory cost will be greater if more storage is required (larger S) or if the memory has to be faster (larger M, R). Similarly, the CPU cost will be larger if the processor has to be faster (larger M, R). The relationship between memory CPU speed and cost is taken as Speed (in MIPS)=k×cost^g, where k and g are empirically derived constants.

The other architecture attribute that is used in the cost computations is the availability of software tools to aid in developing software for MCF systems. This was established by the Selection Committee by defining a menu of support software (i.e., compilers, editors, etc.) required for military applications and then evaluating the relative percentage of this menu available for each architecture. Using data from actual system developments, a curve was generated relating this relative availability of software tools to the cost (per line of code) of developing software for an architecture. This data was ultimately used in the computation of life cycle software costs. Detailed description of the cost analyses described herein can be found in Reference 5.

The Models

Two models of military computer life cycle resource requirements are used in conjunction with the basic methodology described above. These are called the Top Down (TD) and Bottom Up (BU) models. It is frequently useful when building models for forecasting economic data to apply two different approaches in order to cross check the results.

The TD model predicts computer resource requirements by extrapolating trends in overall expenditures and requirements in DoD for various aspects of computer hardware and software. This model predicts, year-by-year, up to 1990 what these expenditures will be for each architecture and then provides relative architecture costs based on the cumulative costs.

The BU model predicts computer resource requirements by using the hardware and software design characteristics of fifteen actual military data processing systems in development or to be developed. The costs of developing all these systems are computed, given each candidate architecture, for the years 1976 and 1985.

The reader will notice that the notations in the Bottom-Up and Top-Down sections of this paper do not always correspond. The two models and analyses were generated and authored independently, except for some initial exchange of ideas and discussion of input data. We have not attempted to normalize the descriptions of the two models.

BOTTOM-UP MODEL

Basic approach

Overview

The computer resource life-cycle cost was estimated for each of 15 Army embedded-computer systems (shown in Table II), which are currently in various phases (conceptual through deployment) of their life cycles. Estimates were made for 1976 and for 1985 production procurements. The lowest cost CFA for each system and for all systems was selected for 1976 and 1985 procurements.

Total Life Cycle Cost

The computer resource life cycle cost for system i and architecture j is defined as:

\[ C_{ij} = HW_{ij} + ASW_{ij} \]  

where

- \( HW_{ij} \) = hardware life cycle cost
- \( ASW_{ij} \) = applications software life cycle cost

Hardware Life Cycle Cost (LCC)

The computer hardware life-cycle cost for a given system using a specific CFA is defined as

\[ HW_{ij} = n_L(i) (P_{ij} + MM_{ij} + SM_{ij}) \]  

where

- \( i \) = index of the system
- \( j \) = index of the architecture
- \( n_L(i) \) = number of units to be produced for system i
- \( L \) = hardware life-cycle cost factor, i.e., ratio of total hardware life-cycle cost to hardware acquisition cost. This factor is assumed to be 2 for a 10-year life cycle
- \( P_{ij} \) = processor acquisition cost
- \( MM_{ij} \) = main memory acquisition cost
- \( SM_{ij} \) = secondary memory acquisition cost

The processor acquisition cost for system i using architecture j is defined as

\[ P_{ij} = K(a_{ij}M_{ij})^{0.4} \]  

where

- \( K \) = constant relating to processor cost
- \( a_{ij} \) = processor speed ratio
- \( M_{ij} \) = operating speed in millions of instructions per second (MIPS)

Equation (3) follows from a commonly cited relationship between performance and cost, namely performance=con-
stant x cost. For the purpose of the BU model g was assumed to be 2.5. To obtain a value for K, in Eq. (3), we used the fact that recent cost/speed data for several military processors seems to indicate that speeds of 0.5 MIPS and processor costs of $48,000 are representative values; consequently

\[ K = 48 \times 10^4 / (5)^4 = 6.3 \times 10^4. \]

This value of K is used in subsequent calculations for 1976 processor cost estimates and is reduced by a factor of 10 for 1985 processor cost estimates based on an assessment of hardware cost reduction over the next decade.

The main memory acquisition cost for system i using architecture j is defined as:

\[ MM_i = c_b(b_j)^P M_i + D M_i \]  \hspace{1cm} (4)

where:

- \( MM_i \) = main memory acquisition cost (in dollars)
- \( b_j \) = static storage ratio
- \( P M_i \) = main memory (in bits) required for program storage in system i; \( M_i \) is derived from system proponents; \( P \) is estimated fraction of \( M_i \) dedicated to program storage vs data storage. (See Table I for values.)
- \( D M_i \) = main memory (in bits) required for data storage in system i; \( M_i \) is derived from system proponents; \( D \) is estimated fraction of \( M_i \) dedicated to data storage vs. program storage
- \( c_b \) = cost per bit of main memory derived from the study of Air Force ADP requirements through the 1980's and Turn's data in Computers in the 1980's. Examination of the price per bit of current militarized memory systems indicates an average cost of 4 cents per bit; i.e., $5000 per 16K byte memory module. This value is used in 1976 cost estimates; 0.4 cents is assumed in 1985 cost estimates.

The secondary memory acquisition cost for system i using architecture j is defined as:

\[ SM_j = C_s(b_j)^P M_a + D'M_a \]  \hspace{1cm} (5)

where:

- \( SM_j \) = secondary memory acquisition cost (in dollars)
- \( b_j \) = static storage ratio
- \( P M_a \) = secondary memory (in bits) required for program storage in system i; \( M_a \) is derived from system proponents while \( P \) is the estimated fraction of \( M_a \) used for program storage vs. data storage
- \( D'M_a \) = secondary memory (in bits) required for data storage in system i; \( M_a \) is derived from system proponents while \( D' \) is the estimated fraction of secondary memory used for data storage vs. program storage
- \( C_s \) = cost per bit of secondary memory derived from References 8 and 9.

Examination of the price per bit of current militarized disc systems indicates an average cost of 0.2 cents per bit, e.g., a 36M bit disc system at $72,000. This value is used in 1976 cost estimates; a cost reduction of 10:1 in the next 10 years is assumed so that a price of 0.02 cents per bit is used in 1985 cost estimates.

To obtain values of \( n_i, M_rj, M_j, M_a, S_j \), a letter was sent from ECOM to project managers of the fifteen systems requesting values for their particular system. Their responses are tabulated in Table I.

The processor speed ratio \( (a_{ij}) \) and static storage ratio \( (b_{ij}) \) attempt to capture the ability of the j-th architecture relative to system i. They are derived by measuring the performance of the three architectures on twelve test programs and by estimating the relative importance, or weight, of each of these programs in computations characteristic of each system. The performance of each architecture on the test programs is summarized in what are called the S, M, and R measures, where:

- \( S_{ik} \) = a measure of the amount of memory (in 8-bit bytes) needed to represent test program k on architecture j.
- \( M_{ik} \) = a measure of the processor/memory transfers required to execute test program k when using architecture j.
- \( R_{ik} \) = a measure (in 8-bit bytes) of the number of internal register-to-register transfers required by the processor to execute test program k on architecture j.

See Reference 10 for details of the test program experiment.

The relevance of the k-th test program to the i-th system is given by the factors \( W_{ik} \), which were obtained by first dividing the twelve programs into two categories: programs that relate principally to I/O, and those that are associated with traditional processor/memory functions. Within these categories, varying degrees of functional overlap occur among the test programs. Initially a gross value was estimated for each of the two categories; subsequently, this value was distributed across the programs of the category. As an example of how the weights were distributed, one data management system resembles contemporary commercial data processing in its use of COBOL and data bases. Thus, its weighting was heavily biased toward I/O and processing routines such as "Linked List Insertion," and lightly biased toward mathematical oriented routines such as "Runge Kutta." Another system, however, is designed primarily for trajectory computations and thus its weighting favored the "Runge Kutta" and was biased against "Linked List Insertion." The sum of all the test programs weights for a given system is unity.

The processor speed ratio \( (a_{ij}) \) and the static storage ratio \( (b_{ij}) \) were obtained by combining the above quantities in the
TABLE I—System Proponent Data

<table>
<thead>
<tr>
<th>Sys. #</th>
<th>System Mission</th>
<th>( n_i )</th>
<th>( \frac{M_j}{(MIPS)} )</th>
<th>*PM (_i)</th>
<th>*DM (_i)</th>
<th>*PMa (_i)</th>
<th>*DMA (_i)</th>
<th>( S_i^{**} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Medium Search</td>
<td>192</td>
<td>1.33</td>
<td>.414</td>
<td>.101</td>
<td>1.9</td>
<td>7.7</td>
<td>32</td>
</tr>
<tr>
<td>2</td>
<td>Medium Command &amp; Control</td>
<td>27</td>
<td>.26</td>
<td>1.638</td>
<td>.412</td>
<td>2.2</td>
<td>8.8</td>
<td>144</td>
</tr>
<tr>
<td>3</td>
<td>Small Search</td>
<td>100</td>
<td>1.00</td>
<td>.512</td>
<td>.128</td>
<td>2.8</td>
<td>11.2</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>Large Command &amp; Control</td>
<td>178</td>
<td>.20</td>
<td>13.600</td>
<td>3.400</td>
<td>4.0</td>
<td>16.0</td>
<td>375</td>
</tr>
<tr>
<td>5</td>
<td>Medium Command &amp; Control</td>
<td>64</td>
<td>.50</td>
<td>1.600</td>
<td>.400</td>
<td>15.8</td>
<td>63.2</td>
<td>47</td>
</tr>
<tr>
<td>6</td>
<td>Large Command &amp; Control</td>
<td>30</td>
<td>.40</td>
<td>3.321</td>
<td>.820</td>
<td>8.4</td>
<td>33.6</td>
<td>250</td>
</tr>
<tr>
<td>7</td>
<td>Small Command &amp; Control</td>
<td>832</td>
<td>.75</td>
<td>.618</td>
<td>.152</td>
<td>.4</td>
<td>1.6</td>
<td>100</td>
</tr>
<tr>
<td>8</td>
<td>Large Communications</td>
<td>616</td>
<td>.18</td>
<td>3.200</td>
<td>.800</td>
<td>13.4</td>
<td>52.0</td>
<td>175</td>
</tr>
<tr>
<td>9</td>
<td>Small Communications</td>
<td>800</td>
<td>.16</td>
<td>.116</td>
<td>.031</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>Small Communications</td>
<td>9</td>
<td>.53</td>
<td>.408</td>
<td>.102</td>
<td>3.2</td>
<td>12.8</td>
<td>83</td>
</tr>
<tr>
<td>11</td>
<td>Small Special Purpose</td>
<td>30</td>
<td>.48</td>
<td>.328</td>
<td>.082</td>
<td>.1</td>
<td>.3</td>
<td>14</td>
</tr>
<tr>
<td>12</td>
<td>Large Data Management</td>
<td>16</td>
<td>.02</td>
<td>1.600</td>
<td>.400</td>
<td>573.4</td>
<td>2293.6</td>
<td>324</td>
</tr>
<tr>
<td>13</td>
<td>Medium Search</td>
<td>50</td>
<td>.35</td>
<td>3.712</td>
<td>.928</td>
<td>3.2</td>
<td>12.8</td>
<td>28</td>
</tr>
<tr>
<td>14</td>
<td>Medium Data Management</td>
<td>8</td>
<td>.80</td>
<td>3.200</td>
<td>.800</td>
<td>1912.0</td>
<td>7648.0</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>Small Guidance &amp; Control</td>
<td>3325</td>
<td>.20</td>
<td>.006</td>
<td>.002</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

* P and D are fractions applied to the proponent's stated memory requirements which reflect an estimate of memory used for programs (P) vs. data (D). Values are expressed in megabits.

** \( S_i \) stated in \( 10^3 \) instructions.
following manner:

\[
\begin{align*}
  a_{ij} &= \frac{\sum_{k=1}^{12} W_{ik}(3M_{ij}+R_{ik})}{\sum_{m=1}^{12} \sum_{k=1}^{12} W_{ik}(3M_{km}+R_{km})} \\
  b_{ij} &= \frac{\sum_{k=1}^{12} W_{ik}S_{ij}}{\sum_{m=1}^{12} \sum_{k=1}^{12} W_{ik}S_{km}}
\end{align*}
\]

(6) (7)

Applications Software Life Cycle Cost

The applications software life cycle cost for system \( i \) using architecture \( j \) is defined as:

\[
\text{AS}_{wi} = C_s S_i L_s
\]

(8)

where

- \( C_s \): cost (in dollars) per instruction of applications software for architecture \( j \)
- \( S_i \): applications software size (in instructions), derived from the system proponents data (Table I).
- \( L_s \): applications software life cycle cost factor, i.e., ratio of applications software life-cycle cost to initial acquisition cost

The software life-cycle cost factor was taken basically from Fisher's report\(^{11}\) which places modifications and retrofits to software at four to five times the cost of the initial product. Thus by taking the midpoint and adding the initial cost as one, we have a value of \( L_s = 5.5 \). The parameter, \( C_s \), the cost per instruction of application software for architecture \( j \), is based on the experience of System Development Corporation with five large scale (24K instructions to 500K instructions) software efforts. The data was compiled by sending questionnaires to the program managers. Program managers responded with: the cost of software production, the number of instructions produced, and for thirteen software tools they estimated what would be the percent increase in project cost if the tool were not available and how much less the project cost would be if the ideal tool were available. From generalizations of this data, it was possible to construct the curves shown in Figure 1. These curves show the variation of cost per instruction as a function of the Tool Availability Index (TAI) for three conditions: (a) worst case, (b) best case and (c) derived median. It should be recognized that the results are largely judgmental and that examples can also be found which yield costs per instruction above and below the worst and best case curves of Figure 1. The Tool Availability Index (TAI) is defined as the ratio of available software tools to the ideal set of software tools. The “ideal set” is defined in a separate report by the Software Evaluation Methodology Subcommittee.\(^{12}\)

By knowing the TAI for a given architecture for any point in time, one can obtain from Figure 1 an estimated cost per instruction. As the percentage of available software tools increases, the cost per instruction of application software can be seen to diminish. The 1976 TAI for each CFA finalist was derived from the report of the Software Evaluation Committee\(^{13}\) as 34%, 50% and 73% for the Interdata 8/32, DEC PDP-11 and IBM S/370 architectures, respectively.* The average of these values is 52% and is indicated graphically in Figure 1. The cost per instruction for the average TAI is approximately $25, $10 and $17 for conditions A, B and C, respectively. The cost per instruction for the 3 CFA finalists in 1976 was estimated at $24, $18, and $12 for Interdata 8/32, DEC PDP-11 and IBM S/370, based upon TAI values of 34%, 50% and 73%, respectively. The corresponding values for 1985 were computed by assuming an annual expenditure of $2M to augment the support software base of the selected CFA. The resulting TAI values in 1985 are 83%, 100% and 100% corresponding to cost per instruction of $10, $7.50 and $7.50 for Interdata 8/32, DEC PDP-11 and IBM S/370, respectively. This data was then employed to compute applications software life cycle cost for each of the 15 systems under consideration, for each CFA and for 1976 and 1985.

Results

Total Life Cycle Costs

Table II illustrates the total life cycle costs for each system and for each of the three architectures for 1985. Circled values indicate the least-cost architecture for a particular system and the least total cost for implementing all fifteen systems. The number of times an architecture is selected is also totaled and shown at the bottom of the chart as the Number of System Preferences. Table II indicates that:

* A subsequent refinement of the data in the SEC report changes these percentages to 37%, 54% and 77%. The changes proved to have no significant impact on the data described herein.
TABLE II.—Total Life Cycle Cost vs CFA
1985 Procurement

<table>
<thead>
<tr>
<th>SYS #</th>
<th>SYSTEM MISSION</th>
<th>n</th>
<th>INTERDATA 8/32</th>
<th>DEC PDP-11</th>
<th>IBM $370</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>HDW</td>
<td>ASW</td>
<td>TOTAL</td>
</tr>
<tr>
<td>1</td>
<td>Medium Search</td>
<td>192</td>
<td>3.9</td>
<td>1.8</td>
<td>5.7</td>
</tr>
<tr>
<td>2</td>
<td>Medium Comm’d and Ctrl</td>
<td>27</td>
<td>0.7</td>
<td>7.9</td>
<td>8.6</td>
</tr>
<tr>
<td>3</td>
<td>Small Search</td>
<td>100</td>
<td>2.2</td>
<td>1.1</td>
<td>3.3</td>
</tr>
<tr>
<td>4</td>
<td>Large Comm’d and Ctrl</td>
<td>178</td>
<td>23.0</td>
<td>20.6</td>
<td>43.6</td>
</tr>
<tr>
<td>5</td>
<td>Medium Comm’d and Ctrl</td>
<td>64</td>
<td>3.4</td>
<td>2.6</td>
<td>6.0</td>
</tr>
<tr>
<td>6</td>
<td>Large Comm’d and Ctrl</td>
<td>30</td>
<td>1.6</td>
<td>13.8</td>
<td>15.4</td>
</tr>
<tr>
<td>7</td>
<td>Small Comm’d and Ctrl</td>
<td>832</td>
<td>13.4</td>
<td>5.5</td>
<td>18.9</td>
</tr>
<tr>
<td>8</td>
<td>Large Communications</td>
<td>616</td>
<td>36.3</td>
<td>9.6</td>
<td>45.9</td>
</tr>
<tr>
<td>9</td>
<td>Small Communications</td>
<td>800</td>
<td>5.5</td>
<td>0.4</td>
<td>5.9</td>
</tr>
<tr>
<td>10</td>
<td>Small Communications</td>
<td>9</td>
<td>0.2</td>
<td>4.6</td>
<td>4.8</td>
</tr>
<tr>
<td>11</td>
<td>Small Special Purpose</td>
<td>30</td>
<td>0.5</td>
<td>0.8</td>
<td>1.3</td>
</tr>
<tr>
<td>12</td>
<td>Large Data Management</td>
<td>16</td>
<td>18.0</td>
<td>17.8</td>
<td>35.8</td>
</tr>
<tr>
<td>13</td>
<td>Medium Search</td>
<td>50</td>
<td>2.4</td>
<td>1.5</td>
<td>3.9</td>
</tr>
<tr>
<td>14</td>
<td>Medium Data Management</td>
<td>8</td>
<td>29.9</td>
<td>1.9</td>
<td>31.8</td>
</tr>
<tr>
<td>15</td>
<td>Small Guidance &amp; Ctrl</td>
<td>3325</td>
<td>203</td>
<td>0.1</td>
<td>20.4</td>
</tr>
<tr>
<td></td>
<td>Total Cost</td>
<td></td>
<td>161</td>
<td>90</td>
<td>251</td>
</tr>
<tr>
<td># System Preferences</td>
<td></td>
<td></td>
<td>14.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The average total life cycle cost for all systems/architectures is $250M in 1985. The 8/32 cost is approximately equal to the average cost while the PDP-11 and S/370 costs are 8.9% below and 8.7% above, respectively, the average cost.

Sensitivity Analysis

Some of the parametric values leading to the 1976 and 1985 results were perturbed in order to determine the sensitivity of the life cycle cost model. The results are described in the following subparagraphs.

When the average cost per instruction in 1976 was doubled to $34, or $9 in excess of the worst-case curve in Figure 1, the resulting hardware/software ratios were 6:1 and 1:2:1 in 1976 and 1985 respectively. The resulting life cycle costs for 1976 and 1985 indicated that, in terms of the comparative costs of the three architectures, the model is relatively insensitive to doubling the cost per applications software instruction. Comparing results, IBM picked up one system implementation at both DEC and Interdata expense in 1976 so that DEC, IBM and Interdata architectures would be preferred in 10.5, 4.0 and 0.5 systems, respectively. The DEC architecture remains the lowest in total life-cycle cost for all systems. For 1985 there were no changes, DEC architecture being preferred in 14.5 systems.

The data shown in Table II reflect an investment in the software bases of the three architectures of $2M annually. The data for 1985 were re-examined to determine the impact of reducing the investment in the software base to $1M annually for two different situations: $17 per instruction of application software and then $34 per instruction. At $17 per instruction the selection of IBM would increase to 5 systems while DEC would drop to 10. This is also predictable since at a $1M per year investment rate in the software base IBM retains for a longer time its cost advantage in producing application software. At this lower investment rate, while IBM has achieved the ideal support software set, DEC has only 83% while Interdata has 61%.

Doubling the cost per application software instruction merely extends a software favorable situation, in that the number of IBM selections equals the number of DEC selections at seven.

The data was re-examined to determine the impact on the results when the S, M, and R measures were used without the discriminating weights applied as described earlier. The results showed that for 1985, the model is relatively insensitive to the weighting factors applied to the S, M, and R values. At $17 per application software instruction a few systems shift from DEC to Interdata; at $34 per instruction a few systems shift from DEC to IBM. DEC remains predominant.

Conclusions

The results obtained with the bottom-up life-cycle cost model are summarized in Table III.

Hence from the bottom-up model results we conclude that the DEC PDP-11 architecture would provide the lowest life-cycle cost for most of the 15 Army embedded-computer
TABLE III—Summary: Bottom Up Life Cycle Cost Analysis

### A. **AVERAGE TOTAL LIFE CYCLE COSTS (MILLIONS) FOR 15 SYSTEMS**

<table>
<thead>
<tr>
<th>Type</th>
<th>Cost 1976</th>
<th>Cost 1985</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
<td>$1750</td>
<td>$175</td>
</tr>
<tr>
<td>Software</td>
<td>$ 162</td>
<td>$  75</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>$1912</td>
<td>$250</td>
</tr>
<tr>
<td><strong>HDW/SW Ratios</strong></td>
<td>12:1</td>
<td>2.4:1</td>
</tr>
</tbody>
</table>

### B. **1976 ARCHITECTURE COMPARISON**

<table>
<thead>
<tr>
<th>Architecture</th>
<th># System Preferences</th>
<th>Relative Total Cost**</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HDW</td>
<td>SW</td>
</tr>
<tr>
<td>8/32</td>
<td>0.92</td>
<td>1.33</td>
</tr>
<tr>
<td>PDP-11</td>
<td>0.91</td>
<td>1.00</td>
</tr>
<tr>
<td>S/370</td>
<td>1.16</td>
<td>0.67</td>
</tr>
</tbody>
</table>

### C. **1985 ARCHITECTURE COMPARISON**

<table>
<thead>
<tr>
<th>Architecture</th>
<th># System Preferences</th>
<th>Relative Total Cost**</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HDW</td>
<td>SW</td>
</tr>
<tr>
<td>8/32</td>
<td>0.92</td>
<td>1.20</td>
</tr>
<tr>
<td>PDP-11</td>
<td>0.91</td>
<td>0.91</td>
</tr>
<tr>
<td>S/370</td>
<td>1.16</td>
<td>0.91</td>
</tr>
</tbody>
</table>

* Hardware/software ratios are calculated from cost data

** 1.00 denotes average cost
systems considered and for the 15 systems as a whole in 1976 and 1985 under the following conditions:

(a) Applications software cost per instruction of $17 and $34 in 1976. Lower costs are assumed in 1985 as the support software base is augmented.
(b) Support software investment rate of $1M, $2M and $3M per year to 1985.
(c) Architecture test measures of effectiveness (S, M and R) are weighted for each system application.
(d) Hardware cost reduction of 10:1 from 1976 to 1985.
(e) Hardware life cycle cost is twice acquisition cost, software life cycle cost is 5.5 times acquisition cost.

The results shown in Table III are not significantly changed if the average applications software cost per instruction in 1976 is doubled to $34 (thereby decreasing the hardware:software ratio by a factor of 2) or if the annual support software investment for the selected CFA is increased to $3M or decreased to $1M.

THE TOP-DOWN MODEL

The Basic Model

The principal outputs of this model are values, $R_{jk}$, $j=1, \ldots, N_y$, $k=1, \ldots, N_a$, which are the discounted costs of an architecture $k$, relative to a reference architecture, totaled over a period of $j$ years. Here, $N_y$ denotes the maximum time period in years and $N_a$ the number of architectures under examination. An element $R_{jk}$ is called a discounted cumulative cost ratio. If the model yields $R_{jk} = R_{j1}$ for $j=1$ through $j=m$, since it has a lower cumulative cost for that period.

The study described herein examines cumulative costs over periods of one to thirteen years ($N_y = 13$) beginning in 1978. Cumulative costs were calculated up to 1990, $j=13$. Since this study compares only three architectures, we have $N_a = 3$. The index $k=1$ represents the IBM S/370 computer family architecture (the reference architecture), $k=2$ the Digital Equipment Corporation PDP-11 architecture, and $k=3$ the Interdata 8/32 architecture.

The model obtains the yearly architecture-dependent cost ($C_{jk}$) by summing the architecture-dependent hardware costs ($H_{jk}$) and software costs ($S_{jk}$).

$$C_{jk} = S_{jk} + H_{jk}.$$  (9)

The nondiscounted cumulative cost through year $m$ for architecture $k$, $D_{mk}$, is simply the sum of all costs from year 1 through year $m$.

$$D_{mk} = \sum_{j=1}^{m} C_{jk}.$$  (10)

The discounted cumulative cost, $D_{mk}$, on the other hand, takes into account the time value of money. Inflation aside, a dollar today is worth more than a dollar tomorrow. The model multiplies cash flows occurring in a year $j$ by a discount factor, $d_j$, and sums the products,

$$D_{mk} = \sum_{j=1}^{m} C_{jk}d_j.$$  (11)

Each discount factor is an average over the year $j$ of the single-payment present-worth factor. These discount factors are the same as those recommended in References 13, 14 and used in Reference 8, volume V.

Since our immediate interest is only in the relative merits of the three architectures, their cumulative cost ratios provide an adequate and useful measure. In addition, the cost ratios are more useful than absolute costs because they serve to cancel out common and possibly unknown multiplicative factors.

Taking architecture one (the IBM S/370) as the reference architecture, we define the nondiscounted cumulative cost ratio as

$$R_{jk} = \frac{D_{jk}}{D_{j1}}.$$  (12)

and, analogously, the discounted cumulative cost ratio as

$$R_{jk} = \frac{D_{jk}}{D_{j1}}.$$  (13)

Consequently, we have $R_{j1} = R_{j1} = 1$ for $j=1, \ldots, N_y$.

Architecture-Dependent Hardware Costs

The model obtains the architecture-dependent hardware costs ($H_{jk}$) by summing the architecture-dependent processor ($P_{jk}$'), main-memory ($M_{jk}$'), and secondary-memory ($E_{jk}$') expenditures.

$$H_{jk} = P_{jk} + M_{jk} + E_{jk}.$$  (14)

The following paragraphs describe the method of arriving at values for these expenditures.

Total yearly hardware expenditures, $B_j$, $j=1, \ldots, N_y$, for CFA-related military computer systems are key inputs to the model. They include computer, main-memory, secondary-memory, I/O and peripheral devices, and related expenditures.

Military systems typically require years between initiation of development and full deployment. This phasing-in period is estimated to be approximately equal to average system development cycle time, which experience indicates runs between 3 and 10 years. In order to approximate the expenditure levels during this phasing-in period, we assume that the expenditures $B_j$, $j=1, \ldots, N_y$, begin at some predetermined level, $B_1$, increase linearly with time over an initial development period ($B_j$, $j=1, \ldots, N_d$), and then remain constant for the remaining period ($B_j$, $j=N_d+1, \ldots, N_y$). For the purpose of the study, the development period, $N_d$, was estimated to be seven years. The model also includes effects of choosing development periods of five and ten years, assuming the same initial and final hardware expenditure levels. The simplifying assumption of constant base hardware expenditures after an initial 1985.
development period appears to be reasonable because: (1) total ADP expenditures in DoD, when measured in uninflated dollars, have been nearly constant in recent years; (2) the principle of level funding has tended to guide DoD budget allocations; (3) decreasing hardware costs on a per-unit basis have tended to offset increasing (inflated) hardware requirements; and (4) the exact dollar assumption used is less important than its equal-handed application to each of the candidate architectures.

Although they are few and far between, there are some estimates and projections of DoD computer system budgets available today. For our purpose, perhaps the most useful was D. A. Fisher's study of ADP costs in DoD. In this report, Fisher estimated that in FY 1973, DoD spent 6.2 to 8.3 billion dollars on ADP. He found that approximately one third of this amount originated in each service, and that about 16 percent of the total went to computer hardware, 45 percent went to software, and 38 percent to other ADP costs, such as support and supplies, keypunching, and computer operation. Using these figures, we deduce that 1.0 to 1.3 billion dollars went to computer hardware and roughly 2.8 to 3.7 billion dollars went to software. During a private conversation with the authors in May, 1976, he mentioned that his recent studies showed that if one were to divide DoD software costs by application, the major portion (approximately 56 percent) goes for embedded computer systems, i.e., the types of systems the CFA project is designed to influence. Approximately 19 percent goes for administrative data processing applications wherein COBOL is the principal language, and 5 percent goes to scientific applications using most commonly FORTRAN. The other 20 percent goes for other types of applications and indirect costs. Assuming these percentages also applied in FY 1973, this would mean that of the approximately 2.8 to 3.7 billion that went to software, about 1.6 to 2.1 billion went for software for embedded computer systems. If we make the assumption that the software-to-hardware cost ratio for embedded computer systems is approximately the same as that for overall DoD ADP systems, we obtain 0.6 to 0.7 billion dollars for hardware for embedded computer systems. Dividing this by three to obtain an estimate of cost for each service, we obtain 0.2 to 0.23 billion. Of course, not all embedded computer systems will be satisfied by CFA. Making a rough assumption that one fourth would be, we obtain a yearly average hardware expenditure rate for a single service of about 50 million dollars. For the purpose of our study, we took this figure as our nominal hardware expenditure level. The exact expenditure level assumed is immaterial since it is the comparative (not absolute) costs that are being computed. In summary, for most of the cases reported in this study, we assumed the yearly base hardware expenditures will remain constant at fifty-million dollars after linearly increasing over a seven-year development period.

The model assumes that the nominal yearly processor expenditures \( K_s \) are a constant \((u)\) times the yearly base hardware expenditures \( B_j \). \( K_s = uB_j \). \( (15) \) Nominal processor expenditures \( K_s \) comprise the nominal CPU \((P_j)\) and nominal main memory \((M_j)\) expenditures but exclude expenditures for I/O busses, devices, and other peripheral gear whose costs are insensitive to the architecture of the processor, \( K_s = P_j + M_j \). \( (16) \)

We assumed in most cases that the constant \( u \) was 0.5; that is, we took the total yearly CPU and main memory costs to be one-half the overall CFA base hardware expenditures. This assumption compares well with a recent survey of DP budgets. In order to measure the sensitivity of the model to \( u \), we also used values of 0.4 and 0.6 as explained in the next section.

The model assumes the nominal yearly main memory cost \( (M_j) \) is a fraction \( (\alpha) \) of the nominal yearly processor expenditures \( (K_j) \), \( M_j = \alpha K_j \). \( (17) \) used on data on military computer systems found in References 15 and 22, it appears that \( \alpha \) usually lies between 0.5 and 0.8. The value used in most of our calculations was 0.65.

Actual main-memory cost \( (M_{mk}) \), which is architecture dependent, is computed from nominal main-memory cost as follows: \( M_{mk} = p_s s_m a M_j \). \( (18) \) The two coefficients, \( p_s \) and \( s_m \), are included in this equation because the cost of memory depends upon the efficiency with which an architecture stores programs as well as the rate at which it uses memory in executing them. The parameter \( s_m \), the normalized s-measure, is indicative of the memory space required to represent a program when using architecture \( k \). This space includes all the storage required to represent and execute the program exclusive of input/output used by the program (since the same data arrays are used by all candidate architectures). The quantity \( p_s \) in equation (18) is a cost-to-performance coefficient for architecture \( k \) which the model obtains by combining the normalized \( m \) and \( r \)-measures as follows: \( p_s = (V_{rk} + W_{mk})^{1/2} \). \( (19) \) The \( m \)-measure, \( m_s \), is a measure of the traffic between main memory and the central processor that is required to execute a program when using architecture \( k \). The \( r \)-measure, \( r_s \), is a measure of the data traffic internal to the central processor. The exponent \( g \) follows from a general relationship, probably first examined by Grosch, between performance and cost, i.e., performance \( \propto \) cost\(^g\). Rein Turn gives a value of \( g \) between 2 and 3; the SADPR-85 Study Group using more recent data, argues for a lower value of about 1.5. Although this latter value was used in most of our calculations, we also investigated the effects of using values of 1 and 2. The parameters \( V \) and \( W \) are weighting factors for the \( r \)- and \( m \)-measures. Their sum is assumed to be one. Because the values obtained for the \( r \)- and \( m \)-measures are almost equal for each of the candidate
architectures, making $r_k$ insensitive to the values of $V$ and $W$, we took both $V$ and $W$ to be 0.5.

The constant $a$ appears in equation $(18)$ because parts of the main memory are insensitive to computer architecture—in particular, the portion occupied by data arrays. The architecture sensitive fraction, $a$, is called the main-memory static-storage ratio. Most of the calculations assume $a$ equals 0.8. The sensitivity studies, however, also examine the effects of using values of 0.7 and 0.9.

The nominal yearly and architecture-independent central processor expenditures ($P_j$) follow from combining equations $(16)$ and $(17)$,

$$P_j = (1 - a)K_j,$$  

(20)

The actual architecture-dependent central-processor expenditures ($P_{jk}$) are given by:

$$P_{jk} = p_kP_j.$$  

(21)

The model assumes the nominal secondary-memory expenditures ($E_j$) are independent of nominal processor costs ($K_j$) and are a fraction, $v$, of the yearly base hardware expenditures ($B_j$), i.e.,

$$E_j = vB_j.$$  

(22)

From these it obtains the secondary-memory expenditures that are architecture-sensitive by

$$E_{jk} = b_k E_j.$$  

(23)

The constant $b$, the secondary-memory static-storage ratio, is analogous to the main-memory static-storage ratio ($a$), and the parameter, $b_k$, is the normalized s-measure discussed in this section.

Systems described in References 15 and 22 led us to a value of $v$ of 0.1 and $b$ of 0.2. The sensitivity studies discussed in the next section examine the effects of choosing $v$ equal 0.0 and 0.2 and $b$ equal 0.1 and 0.3, and show a relative insensitivity of the model to uncertainties in the values of these parameters.

Architecture-Dependent Software Costs

For architecture $k$, the total yearly software expenditure ($S_{jk}$) is the sum of the yearly support-software expenditure ($Q_j$) plus the architecture-dependent application-software expenditure ($A_jF_{jk}$),

$$S_{jk} = Q_j + A_jF_{jk}.$$  

(24)

We define support software to be software tools such as compilers, loaders, linkers, simulators, assemblers, submonitors, operating systems, and debug packages and we assume that the amount of money allocated for the development of these tools is independent of the architecture chosen. Hence, the model assumes that support software will be developed with an expenditure of $Q_j$ dollars for each year $j$ where $j=1, \ldots, 13$. For the cases considered in this paper, we let value $Q_j$ be constant at $x$ million dollars, for $j=2$ through 13 and $Q_1=0$. The sensitivity studies examined the effects of allowing $x$ to range from zero to eight million dollars in two million dollar increments. Although $x$ is architecture independent, it plays a dominant role in determining the lowest cumulative cost alternative.

The model assumes that base (or nominal) applications-software expenditure in year $j$, denoted $A_j$, is a constant ($\rho$) times the yearly base hardware expenditure, $B_j$,

$$A_j = \rho B_j.$$  

(25)

For lack of a better name, we call $\rho$ the software-to-hardware ratio. Working with the reports of Fisher$^{11}$ and others,$^{8,9,10,11,12}$ we estimate $\rho$ to be about 2.5 to 3 for general-purpose DoD computer systems, and possibly $1/2$ to 2 for embedded systems,$^{16}$ which usually have multiple deployments of the same software and hardware and do not have software bundled into the system price.

The constant $\rho$ is one of the most crucial parameters in the model. Because of its importance and because of the difficulty of estimating its value, we determined the cumulative-cost ratios for 1985 and 1990 using values of $\rho$ of $1/4$, $1/2$, 1, 2, and 4. Figure 2 shows the 1990 results for support-software expenditures ($Q_j$) of two million dollars and clearly illustrates the importance of $\rho$ in determining the most cost-effective computer family architecture.

At this point, a word of caution is in order. The error analysis, to be described later, shows that the expected uncertainties of the values of the input parameters can result in substantial uncertainties in the cumulative cost ratios. They imply that the reader should interpret the PDP-11 and Interdata 8/32 curves of Figure 2 as ribbons having widths on the order of twenty to thirty percent of the illustrated values; the choice of the most desirable architec-

![Figure 2: Support Software Expenditure, $Q_j = 2 \times 10^6$ 1990 Curves](From the collection of the Computer History Museum (www.computerhistory.org))
ture for values of $\rho$ between $\frac{1}{4}$ and $2$ (the values expected in the military computing environment) is by no means clear.

The architecture-dependent applications-software costs, mentioned earlier, are $A_k F_{jk}$. Here $F_{jk}$ is the amount of base (nominal) applications-software expenditure that should be used in determining the actual applications-software cost for architecture $k$. It is dependent upon the available support software for architecture $k$ in year $j$.

In order to derive an expression for $F_{jk}$ we began with the same curves shown in Figure 1 of the Bottom Up model. These curves give the cost per machine language instruction as a function of available support software. One-hundred percent support software means that an "ideal" set of software tools for military computer systems is available. W. Svirsky, T. Giles, and A. Irwin of System Development Corporation, West Long Beach, New Jersey, generated these curves on the basis of the results of a questionnaire sent to SDC project managers of five large scale command and control software efforts. They noted, however, that the curves are largely judgmental and examples can be found that yield costs per instruction above and below the worst and best case data. The absolute cost per instruction does not affect the Top Down model, however, because the model depends only upon the relative cost-per-instruction vs support software, as opposed to the absolute cost per instruction given by the curve.

From a graph of the SDC median curve, it appeared to us that the interpolation function

$$y(s') = y_m + (y_m - y_m)\exp(-s'/c)$$

might provide a reasonably good fit. Here $y$ is the dollars per machine language instruction, $s'$ is the support software availability in percent, and $y_m$, $y_m$, and $c$ are constants. Insisting that this function pass through the three points $(s', y)=(20, 32.5), (50, 17.75)$, and $(80, 10)$, in accordance with the SDC supplied curve, we obtained $y_m = 1.4196$, $y_m = 49.151$ and $c = 46.6171$. Next we assumed that

$$F_{jk} = k'y(s'),$$

where $k'$ is a constant whose value is determined by the additional assumption that when the average of the available support software of the three alternative architectures in year 1, call it $s'$, is substituted in equation $(27)$ for $s'$ we should obtain $F_{jk} = 1$. Here we assume $s'$ can be expressed as

$$s' = \frac{100}{3} \left( \frac{U_{jk}}{T_1} + \frac{U_{12}}{T_2} + \frac{U_{13}}{T_3} \right),$$

where $U_{jk}$ denotes the available support software (measured in dollars) in year $j$ for architecture $k$, and $T_k$ the dollar value of a 100 percent support-software base (or ideal support-software base) for architecture $k$.

Several members of the CFA selection committee surveyed the industry and estimated values for the currently existing support-software base for each candidate architecture and also estimated $T_k$. The values they came up with were:

<table>
<thead>
<tr>
<th>Currently Available Support-Software Base ($)</th>
<th>$T_k$ ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM 370</td>
<td>31,049M</td>
</tr>
<tr>
<td>DEC PDP-11</td>
<td>20,790M</td>
</tr>
<tr>
<td>INT 8/32</td>
<td>14,100M</td>
</tr>
</tbody>
</table>

Making the assumption that the currently available support-software base values given above could be used in the model for $U_{jk}$, $k=1, \ldots, 3$, we obtained from equation $(28)$ that $s'=49.7$, and

$$k' = l/y(s') = 0.05601$$

Expressing $F_{jk}$ as

$$F_{jk} = f_m + (f_m - f_m)\exp(-hU_{jk}/T_k),$$

we have

$$f_m = k'y_m = 0.0795$$

and

$$h = 100/c = 2.1451.$$
Results and sensitivity studies

Figures 2 and 3 summarize the results for 1990 of varying the software-to-hardware ratio ($\rho = 1/6, 1/4, 1/2, 1, 2, 4$) and the support-software expenditures ($Q_j = 0, 1 \times 10^6, 2 \times 10^6, 4 \times 10^6, 8 \times 10^6$). These variations constitute the first thirty cases of the sensitivity studies.

All of the sensitivity studies assume the annual discount rate is ten percent (the value recommended by References 13, 14 and used in Reference 8).

Further assessing the model's sensitivity, we selectively perturbed individual input parameters about case number 18's ($\rho = 1$ and $Q_j = 2 \times 10^6$) input data set to obtain twenty-nine additional input data sets. We chose the case 18 data set as a reference because, at this time, it appears to represent the most reasonable set of data based on our current understanding of the requirements of military tactical computer systems, the candidate architectures, and probable life of the MCF.

Figure 4 summarizes the results of the sensitivity studies (for cases 31 through 59) for 1990. The dotted lines in these figures are indicative of the discounted cumulative cost ratios for the reference data set (case 18), and the arrows show the amount and direction of movement from these values as the result of parameter perturbation. From these figures, we see uncertainties in the percentage of the ideal support-software base available in year 1 ($U_{jk}/T_k$), and uncertainties in $u$, the ratio of yearly processor expenditure to base hardware expenditure ($K_j/B_j$), can have significant impacts on the results.

Error analysis

Making the assumption that the errors in the measurements of the model's parameters are normally distributed, the variance of the mean of the calculated discounted cumulative cost ratio $R_{jk}^*$ due to variances of the means of the model parameters $X_m$, $m = 1, \ldots, N_p$, can be approximated by

$$\sigma_m^2 = \frac{1}{N_p} \sum_{m=1}^{N_p} \left( \frac{\partial R_{jk}^*}{\partial X_m} \right)^2 \sigma_m^2. \quad (35)$$

where $N_m$ is the number of parameters under consideration, and $\sigma_m$ is the standard deviation of the mean of the parameter $X_m$.\textsuperscript{16, pp. 97–98}

The partial derivatives were derived from the results of the sensitivity studies in the preceding section. The standard deviations ($\sigma_m$) were obtained for each parameter by the method of "educated guess." It should be noted that

### Table IV

<table>
<thead>
<tr>
<th>Year</th>
<th>Case 18 (Reference)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>98.3</td>
</tr>
<tr>
<td>2</td>
<td>112.6</td>
</tr>
<tr>
<td>3</td>
<td>137</td>
</tr>
<tr>
<td>4</td>
<td>166</td>
</tr>
<tr>
<td>5</td>
<td>192</td>
</tr>
<tr>
<td>6</td>
<td>21.4</td>
</tr>
<tr>
<td>7</td>
<td>28.6</td>
</tr>
<tr>
<td>8</td>
<td>35.7</td>
</tr>
<tr>
<td>9</td>
<td>42.8</td>
</tr>
<tr>
<td>10</td>
<td>50.0</td>
</tr>
<tr>
<td>11</td>
<td>50.0</td>
</tr>
<tr>
<td>12</td>
<td>50.0</td>
</tr>
<tr>
<td>13</td>
<td>50.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Base Appl. Sftwe Exp.</th>
<th>78</th>
<th>79</th>
<th>80</th>
<th>81</th>
<th>82</th>
<th>83</th>
<th>84</th>
<th>85</th>
<th>86</th>
<th>87</th>
<th>88</th>
<th>89</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Hdw Exp. (SM1)</td>
<td>7.14</td>
<td>14.3</td>
<td>21.4</td>
<td>28.6</td>
<td>35.7</td>
<td>42.8</td>
<td>50.0</td>
<td>50.0</td>
<td>50.0</td>
<td>50.0</td>
<td>50.0</td>
<td>50.0</td>
<td>50.0</td>
</tr>
<tr>
<td>Total Yearly Software</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) IBM (SM1)</td>
<td>4.86</td>
<td>10.9</td>
<td>14.3</td>
<td>17.1</td>
<td>19.4</td>
<td>21.3</td>
<td>22.8</td>
<td>21.3</td>
<td>19.9</td>
<td>18.6</td>
<td>17.5</td>
<td>14.4</td>
<td>15.4</td>
</tr>
<tr>
<td>(2) DEC</td>
<td>7.48</td>
<td>15.7</td>
<td>20.8</td>
<td>24.9</td>
<td>28.2</td>
<td>30.8</td>
<td>32.9</td>
<td>30.4</td>
<td>28.1</td>
<td>26.0</td>
<td>24.2</td>
<td>22.5</td>
<td>21.0</td>
</tr>
<tr>
<td>(3) INT</td>
<td>10.2</td>
<td>20.6</td>
<td>27.4</td>
<td>33.0</td>
<td>37.4</td>
<td>40.8</td>
<td>43.5</td>
<td>40.0</td>
<td>36.8</td>
<td>34.0</td>
<td>31.4</td>
<td>29.0</td>
<td>26.9</td>
</tr>
<tr>
<td>Total Yearly Hardware</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) IBM (SM1)</td>
<td>4.26</td>
<td>8.59</td>
<td>12.8</td>
<td>17.2</td>
<td>21.4</td>
<td>25.7</td>
<td>30.0</td>
<td>30.0</td>
<td>30.0</td>
<td>30.0</td>
<td>30.0</td>
<td>30.0</td>
<td>30.0</td>
</tr>
<tr>
<td>(2) DEC</td>
<td>3.09</td>
<td>6.22</td>
<td>9.32</td>
<td>12.4</td>
<td>15.5</td>
<td>18.6</td>
<td>21.8</td>
<td>21.8</td>
<td>21.8</td>
<td>21.8</td>
<td>21.8</td>
<td>21.8</td>
<td>21.8</td>
</tr>
<tr>
<td>(3) INT</td>
<td>2.58</td>
<td>5.20</td>
<td>7.77</td>
<td>10.4</td>
<td>13.0</td>
<td>15.5</td>
<td>18.2</td>
<td>18.2</td>
<td>18.2</td>
<td>18.2</td>
<td>18.2</td>
<td>18.2</td>
<td>18.2</td>
</tr>
<tr>
<td>Total Discounted Cumulative Cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) IBM (SM1)</td>
<td>8.70</td>
<td>25.6</td>
<td>47.0</td>
<td>71.6</td>
<td>98.3</td>
<td>126</td>
<td>154</td>
<td>179</td>
<td>202</td>
<td>221</td>
<td>239</td>
<td>254</td>
<td>268</td>
</tr>
<tr>
<td>(2) DEC</td>
<td>10.1</td>
<td>29.1</td>
<td>52.8</td>
<td>79.6</td>
<td>108</td>
<td>137</td>
<td>166</td>
<td>192</td>
<td>214</td>
<td>234</td>
<td>250</td>
<td>265</td>
<td>278</td>
</tr>
<tr>
<td>(3) INT</td>
<td>12.2</td>
<td>34.5</td>
<td>62.2</td>
<td>93.3</td>
<td>126</td>
<td>160</td>
<td>192</td>
<td>221</td>
<td>245</td>
<td>266</td>
<td>284</td>
<td>300</td>
<td>314</td>
</tr>
<tr>
<td>Total Discounted Cum. Ratios</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) IBM</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>(2) DEC</td>
<td>1.16</td>
<td>1.14</td>
<td>1.12</td>
<td>1.11</td>
<td>1.10</td>
<td>1.09</td>
<td>1.08</td>
<td>1.07</td>
<td>1.06</td>
<td>1.05</td>
<td>1.04</td>
<td>1.04</td>
<td>1.04</td>
</tr>
<tr>
<td>(3) INT</td>
<td>1.40</td>
<td>1.35</td>
<td>1.32</td>
<td>1.30</td>
<td>1.28</td>
<td>1.26</td>
<td>1.25</td>
<td>1.23</td>
<td>1.21</td>
<td>1.19</td>
<td>1.18</td>
<td>1.17</td>
<td>1.17</td>
</tr>
<tr>
<td>Discount Factor (10%)</td>
<td>0.954</td>
<td>0.867</td>
<td>0.788</td>
<td>0.717</td>
<td>0.652</td>
<td>0.592</td>
<td>0.538</td>
<td>0.499</td>
<td>0.445</td>
<td>0.405</td>
<td>0.368</td>
<td>0.334</td>
<td>0.304</td>
</tr>
</tbody>
</table>

1. $\sigma = 1$  $Q_j = 2 \times 10^6$  Dev. Cycle = 7  $a = 0.65$  $b = 1.5$  $\alpha = 0.8$  $\beta = 0.2$
2. $s_1, s_2, s_3 = 1.208, 1.000, 0.828$  $m_1, m_2, m_3 = 1.266, 0.928, 0.850$  $r_1, r_2, r_3 = 1.292, 0.938, 0.825$
the standard deviations for $p$ and $Q_j$ were assumed to be zero. This is not because they are actually zero (in fact, they are quite large) but because we desired to obtain an estimate of the standard deviation as a function of $p$ and $Q_j$ for Figure 2. Here, $Q_j$ and $p$ are independent variables.

The results of these calculations indicate that the uncertainties in the model's major results are quite large. Table V summarizes the findings for a software-to-hardware ratio of one ($p=1$) and support-software expenditures of two million dollars per year.

These results imply that because of uncertainties in our input data we cannot clearly resolve the question of which architecture is the most cost effective for this software-to-hardware ratio.

**INTERPRETING THE RESULTS OF THE MODELS**

The two models serve as checks against one another. The bottom-up results indicate that in most circumstances the DEC PDP-11 is superior to both the IBM S/370 and Interdata 8/32 architectures. The top-down model results, on the other hand, indicate that the S/370 is superior for high (greater than one) software-to-hardware cost ratios, while the Interdata 8/32 is slightly better for low (less than one-fourth) ratios, and the PDP-11 is best in between. These apparently conflicting results were found to be due to uncertainties in the input data, to different input requirements, to contrasting basic model assumptions, and to different methods of combining the same input data.

For example, the bottom-up model weights the raw $S$, $M$, and $R$ data, provided by CMU for the individual test programs, according to the estimated relevance of each

---

**TABLE V**

<table>
<thead>
<tr>
<th>K</th>
<th>$R_{m}$</th>
<th>$\sigma_{m}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM</td>
<td>1</td>
<td>1.00</td>
</tr>
<tr>
<td>DEC</td>
<td>2</td>
<td>1.06</td>
</tr>
<tr>
<td>INT</td>
<td>3</td>
<td>1.22</td>
</tr>
</tbody>
</table>

---

**1990 DISCOUNTED CUMULATIVE COST RATIOS**

- IBM 370
- DEC PDP-11
- INTERDATA 8/32

---

*From the collection of the Computer History Museum (www.computerhistory.org)*
program in each system application. It then combines these into the composite processor speed and static storage ratios, $a_{ij}$ and $b_{ij}$. The top-down model, on the other hand, uses composite $S$, $M$, $R$ ratios, which were derived by CMU from the individual $S$, $M$, $R$ measures for each test program and which were weighted by CMU to obtain minimum statistical variance in these ratios rather than to reflect the importance of particular application programs. A result of these two different approaches to using the individual test program $S$, $M$, and $R$ data is a difference in the computed architectural efficiency of the Interdata 8/32 as compared to the PDP-II. In the first case they are comparable, in the second the 8/32 is superior. A further result of this difference is that if the unweighted $S$, $M$, and $R$ data are used in the bottom-up model, then the 8/32 becomes the superior architecture in the 1976 calculations when the hardware-software cost ratio is high. This agrees with the top-down model results. Conversely, if the $S$, $M$, and $R$ data used in the top-down model were weighted as in the bottom-up model, better agreement between the models would result.

As another example, the assumptions leading to the ratio of software costs to hardware costs are clearly among the most important to the ultimate results, while at the same time are among the most difficult to support with actual data.

The uncertainty calculations in the preceding section for the top-down model could be applied to the bottom-up model with similar results expected. Because of the size of these uncertainties, the results of the models must be interpreted with caution. By chance, each of the three architectures evaluated had either superior hardware attributes (the 8/32), or superior software attributes (the S/370), or a good combination of the two (the PDP-II). As a result, the combined hardware-software effectiveness of the three architectures were relatively close. Probably the strongest conclusions to be derived from the life cycle cost evaluations are that, within the uncertainties resulting from propagating errors in the input data throughout each model’s calculations, (1) the models agree and (2) all three architectures would be comparable choices based on life-cycle costs.

REFERENCES


